15｜8X Flow（中）：如何通过模型发现业务系统的变化点？

2021-08-05 徐昊

《如何落地业务建模》 课程介绍 >

讲述：徐昊
时长 21:01 大小 19.25M

你好，我是徐昊。今天我们继续 8X Flow 的学习。

上节课我们讲到可以将逻辑分为领域逻辑和业务逻辑，而且可以通过不同的方式，分别对领域逻辑和业务逻辑建模。我们也知道了，领域与运营无关，源自某个特定问题域，而业务逻辑与运营相关，大量的业务逻辑源自运营中实践经验的总结。

这个时候我们就需要考虑一个问题：领域逻辑真的能复用吗？我们来分析一下。

领域逻辑真的能复用吗？

由此我们会发现，业务逻辑不如领域逻辑稳定。毕竟业务关注的问题是如何盈利，而不是怎么准确地描述某个问题域。这似乎和我们长久以来的想法相违背：我们希望通过模型精

确建模某个问题域，从而实现对该问题域的复用。

然而**在商业社会里，复用某个领域，就是围绕这个领域构建能够盈利的运营模式**。也就是说，复用某个领域，除了复用领域功能之外，还要构造运营实体。

就像直到今天不会还有人天真地以为，只要复制了淘宝的功能就能再造一个阿里吧！软件总是简单的，但是构造背后的运营实体，以及确立成本合理的运营模式，才是复用的真正重点和难点。换言之，**业务知识的重建与复用，才是领域复用的关键**。

从这个角度来说，领域驱动设计就是天真的谎言。因为它所畅想的美好的复用方式，大多源自技术领域的领域系统，并且不假思索地将这种复用模式推广到业务系统，而忽略了业务系统的运营特定性与领域中立性，这就给我们植入了一个虚假的愿景：通过领域驱动设计，我们可以构造某个领域的通用产品，正如数据库一样，然后就可以忽略业务逻辑，在不同企业间形成复用。

按照这个思路，领域驱动设计所希冀的最大成就，就是实现类似于 MRP（Manufacturing Resource Planning）或者 ERP（Enterprise Resource Planning）这类领域产品。然而，事实告诉我们，所有 ERP 系统最终都会走上两条路。

1. 保持产品不变，通过管理咨询公司介入，重组企业流程，使得它们的运营模式可以与产品匹配。
2. 大量定制产品特性，将受众公司的业务逻辑嵌入到领域产品中。

所以就算把领域做成了产品，也还是不能离开业务逻辑直接复用。要么改变客户的业务以适应产品，要么再把业务逻辑加回到产品中来。所以**领域驱动设计其实是个非常纠结且过时的理念**。领域驱动中的大部分不协调和奇怪之处，都不是另有深意，而只是单纯地想错了。

再顺便说一句，知识消化的“两关联一循环”其实是尝试去解决这些没有从领域逻辑中分离开的那部分业务逻辑的。**毕竟单纯只是建模领域逻辑的话，找到一个简洁的隐喻，将系统核心问题描述清楚，比其他一切手段都有用，也都重要**。

要是实在找不到，通过知识消化磨出一个对象模型，也凑合能用。这也是为什么我说用知识消化对领域逻辑建模，没有对业务逻辑建模来得有效（也可以用）的原因。

那么对于业务系统，也就是与盈利相关的系统，我们更应该关注的是业务逻辑（将其中的领域逻辑做好隔离就行）。那么对于业务逻辑，我们有什么好办法去系统化地理解吗？

答案是有！

现实生活中，**在我们现行法律下，所有企业活动都必须在合同法（现在是民法典）的框架之下进行，而且需要满足第三方审计要求**。所以就有这么一群人，他们可以在完全不了解某公司领域的前提下，能快速且充分地了解该公司的业务情况，并指出其中可能存在的问题。这些人就是财务审计、合同审计和职业经理人。

那么也正是这么一群人，每天都在完成我们眼中的艰深的工作：**虽然对具体情况了解不多，但能快速且细致地理解业务逻辑**。于是，我们只要学习他们是怎么做的，就可以将他们的方法用于理解业务逻辑，并进行业务建模。

他们是怎么做的呢？也就是：以凭证追溯从财务角度理解企业经营，以及从合同履约理解企业业务。

事实上，构成审计核心逻辑的凭证追溯，也构成了四色建模的核心逻辑。而**合同履约则是 8X Flow 的核心逻辑**。

从合同履约理解业务

所有业务活动都需要在合同（具有法律效力的口头约定）的框架下进行，这是法律法规的要求。

比如说，我们网上购物之所以能拿到货品，是因为我们和网店之间存在采购合同（也就是订单），其中规定了为顾客提供货品是商家的义务，拿到货品则是我们的权利。同样的，商家之所以能收到钱，也是因为采购合同的存在，付款是我们的义务，收款则是商家的权利。

以上是权责正常得以行使的情况。但是除此之外，还有大量的业务逻辑是由违约场景构成的。比如还是网购的例子，如果我们没有付款，那么在一定时效内，订单可能会关闭。如果是预定订单，那么商家就有权利不退还预付款等等。

违约场景同样也是合同上下文中规定的权责项。就是说，虽然一方违约，需要承担对应的责任，但是这个责任也需要预先约定，由合同覆盖。

不过如果违约方对于违约责任的承担继续出现了违约呢？比如欠款不还，触发罚息，罚息继续不还，触发强制要求本金归还。或者如果再继续不还呢？那只有去法院打官司了。这时候法院会根据合同中约定的权责履约情况进行裁定，也就是**终极履约**。

于是我们发现，在所有的业务逻辑中，**权责履约是最小的业务交互，合同是最小的业务上下文**。那么我们就能使用**权责和合同上下文**对业务进行建模了。

对于权责履约，我们可以使用履约请求（表示一个时间段）和履约确认（时间点）这样的结构来表示。并将相应的凭证与履约请求关联，以表示发起履约的依据。同样，履约确认也会与相应的凭证关联，以表示履约的证明。

履约请求是由权利方发起的，并要求义务方在规定时限内完成履约。比如我们在网上购物，完成下单之后去支付。这个过程从合同角度看，是权利方卖家要求义务方买家，在 15 分钟内完成对订单支付的履约。如果买家在规定时间内完成支付，就是履约成功。否则就是履约失败，比如没有支付或者因为网络故障等情况，未完成支付。

凭证在权责履约的范围内，可以按照四色建模的方式寻找。也就是说，**针对每一个权责履约，实际是在业务上定义了一小段时间线，从履约请求开始，到履约确认为止**。这是利用了事件建模中的多时间线法。而我们可以在这条时间线上追溯履约中的关键凭证。

除了履约时间线上的凭证之外，履约请求还可以引用合同上下文中的其他凭证。**特别是，履约请求和履约确认本身也是凭证，可以供其他请求与确认引用**。比如，卖家未在约定时间内发货，那么违约取消合同的时候，就可以引用之前的支付履约请求或者支付履约确认，证明卖家没有发货。

明确了履约的结构之后，合同可以作为上下文对象，用以涵盖对应的所有履约项。而且由于合同只存在于双方之间（多方合同是多份双方合同），因而任何合同上下文中都存在两个角色对象。比如采购订单合同中的买家和卖家，快递合同中的邮寄方和承运方等。

在整个合同上下文中，权责也是围绕着两个角色展开的。**所以，我们可以将合同上下文看做两个角色间业务交互的证据的聚合。这是一种业务上存在的聚合关系，是一种比对象聚**

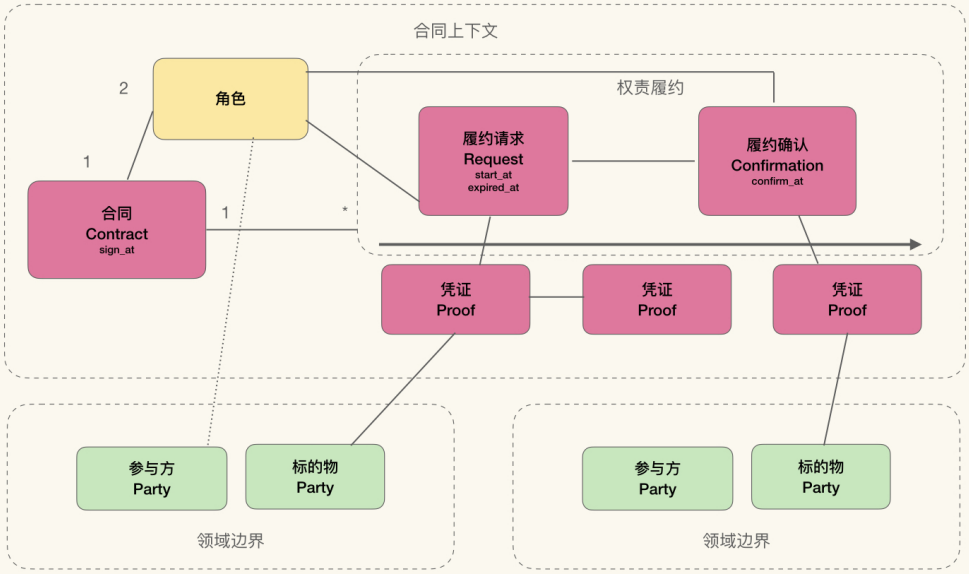
合更具有业务含义的包含关系。

合同的参与方以及凭证中的标的物，可能来自领域系统。比如说，我们网购的采购合同中的支付凭证，牵扯到的标的物就是商品。那么我们可以将商品看做领域系统产品目录中的概念，然后通过凭证引用领域系统中的概念，让领域系统中的概念参与到业务逻辑中来。

到这里，8X Flow 的核心概念就讲完了，接下来我们看看该怎么通过 8X Flow 进行建模。

使用 8X Flow 建模

如下图所示为 8X Flow 的元模型（Meta Model）图，表示了上面我们所讲的核心概念，以及它们之间的关联：



使用 8X Flow 建模的流程大致是这样的：

1. 寻找合同上下文，明确合同的参与方；
2. 寻找合同中的主要履约项，按四色建模寻找凭证；
3. 对于主要履约项，寻找违约情况，设立新履约项，按四色建模寻找凭证；
4. 重复 3，直到不得不打官司为止；
5. 将参与方和标的物划分入领域边界。

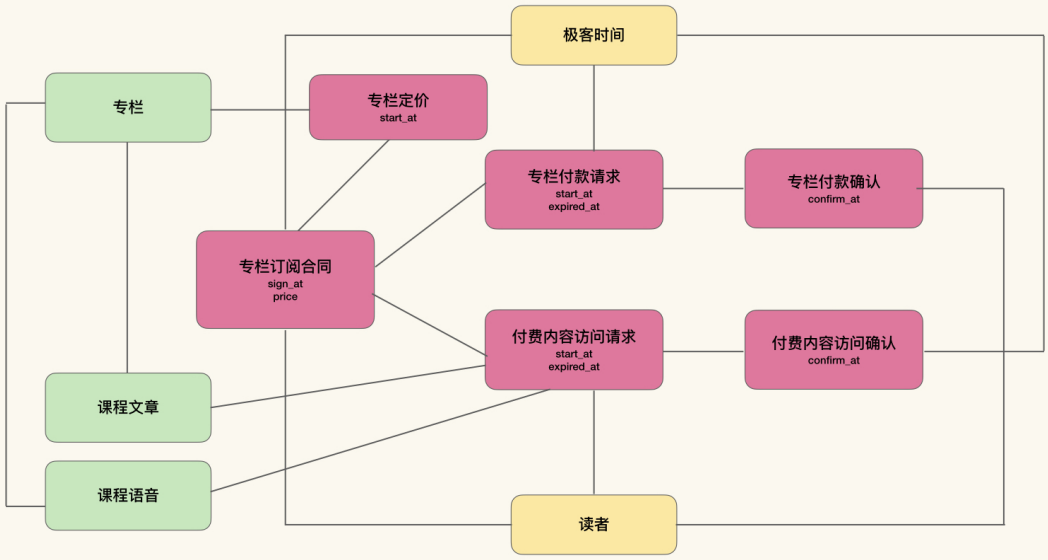
下面就通过例子来看一下这个过程，还是我们的老朋友，极客时间专栏。首先来看一下读者侧，合同是非常明确的，即专栏订阅合同。参与方就是读者和极客时间。

然后，我们来看看主要履约项：

支付订阅费用，权利方是极客时间，义务方是读者。

访问付费内容，权利方是读者，义务方是极客时间。

那么我们可以建立一个业务模型，里面包含合同和这两个主要履约项：



接下来我们就需要寻找违约情况了，对应上面讲的两个权责，那么分别存在如下两种违约情况：

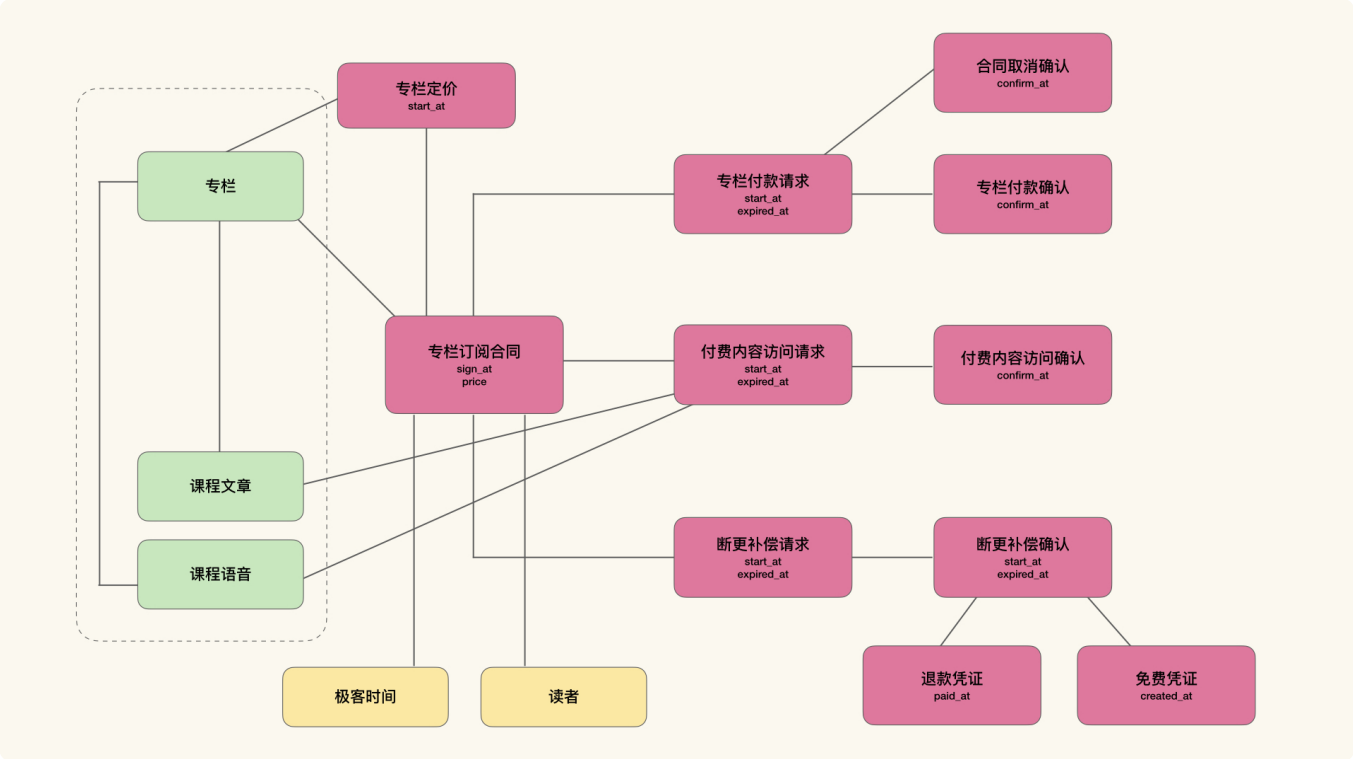
未在规定时间内完成支付，那么合同自动作废，读者并不承担额外责任。权利方是极客时间，义务方是读者。

如果专栏出现断更，没能按说明提供内容，专栏下架，极客时间退钱，并且在下次同专栏上架时，不再向原读者收取任何费用。权利方是读者，义务方是极客时间。

对于第一项违约，合同作废是自动行为，不需要义务方确认，所以不需要有独立的权责项，也没有后续继续违约的操作了。

对于第二项违约，如果极客时间拒不退钱，那么读者也只能走法律手段了。这就已经触碰到需要走法律程序的边界了，那我们就不需要继续寻找违约项了。

所以我们可以继续添加模型到上面的模型图中，并将标的物划入不同的上下文，以表示领域边界，如下图所示。这里我也说明一下，由于图片大小关系，我省略了一些凭证，还简化了权责方与请求和确认的关联。不过这并不影响对模型的整体理解，你可以着重看一下权责履约项：



至此，我们就通过 8X Flow 完成了对极客时间专栏的建模。不过如果我们仔细观察得到的模型，会发现几个有意思的现象。

首先，**通过 Request-Confirmation 表示的履约结构是一种异步结构：在规定时限内，未得到确认之前，履约处在未知状态。这种异步并不是技术上的刻意选择，而是业务的真实状态。**

让我们回想一下真实世界中的业务操作，其实都是这样的异步状态：权利方主张，义务方履约。而我们在软件世界中习惯的是同步状态，是在很短时间内完成的主张 - 履约。但无论这个完成的时间有多短，在概念上，主张、履约仍是异步结构。

所以严格意义来说，我们过往通过同步方式建模都是错的，都是从实现方式去理解问题，而不是从问题本身出发。

其次，**如果履约出现错误，也就是不能履约的情况，那么对于这种异常状态的修正，就需要触发新的履约过程**。

想想看，在现实世界中的业务，从来不存在可以自动修正的情况。因为任何履约失败，都存在破坏合同条款，最终引起法律纠纷的可能。所以不可能在合同双方不知情，也没有提前协商的情况下，对未完成的履约项进行修正（而且严格意义来讲，这么做也是违法的）。

因而，**从业务角度出发，履约中的异常会触发新的履约项，从而在合同的上下文中维持业务的一致性**。

也就是说，如果我们完全从业务出发，不受对象模型搅合的话，就会自然地得到异步为主的模型以表示业务逻辑。那么接下来，就让我们看看合同上下文。

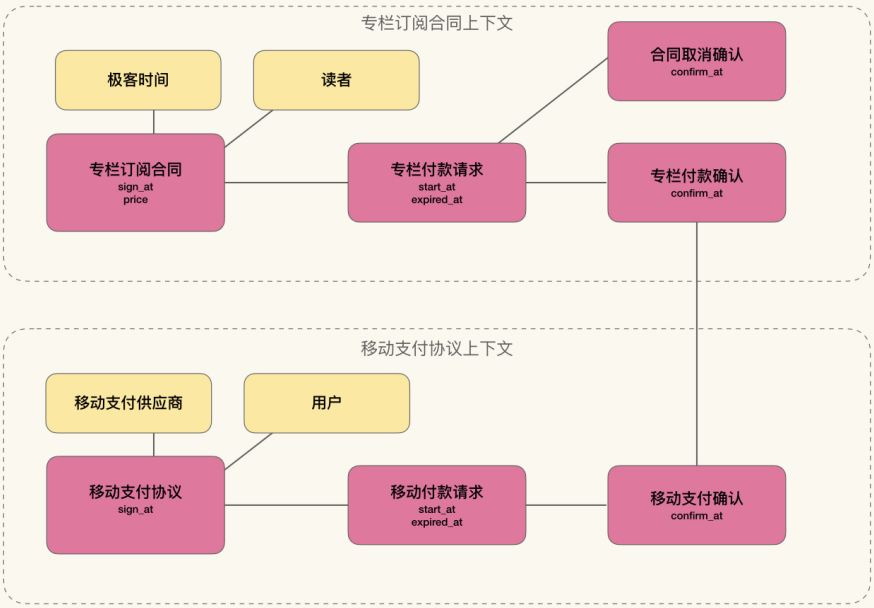
凭证角色化建立合同间的关联

如果再仔细看前面的模型，你可能会发现，我们将支付部分归属于专栏订阅合同了。这表明什么呢？**表明这是一个线下的现金交易模型。**也就是读者和极客时间面对面完成了现金交易，然后极客时间开具发票。

因为在专栏订阅的合同上下文中，参与的双方是读者和极客时间，既没有微信，也没有其他的参与方。**因而说明，这是个现金交易模式**。不过，这显然不适合现在的数字化时代。那么如果我们需要支持移动支付，模型会有什么改变呢？

首当其冲的改变就是引入了第三个角色：**移动支付供应商**。我们知道合同只能在两者之间签署，那么我们势必需要引入另一个合同上下文，去表示移动支付供应商和读者之间的关系。也就是说，在读者和移动支付供应商供应商之间，存在一个合同，用以支撑支付行为。

这个合同呢，其实你也不陌生，就是你开通微信或者支付宝支付服务时，签署的用户协议（对，就是你看也没看就点了“同意”的那篇“废话”）。在这个用户协议中，规定了如何付款与收款。那么我们同样可以通过 8X Flow 对这个用户协议进行建模，过程与上面大同小异，不再赘述，直接来看结果：



通过这个模型我们很明显就能发现，**业务模式改变了，从面对面的现金交易，变成了借由第三方支付服务完成的交易。**

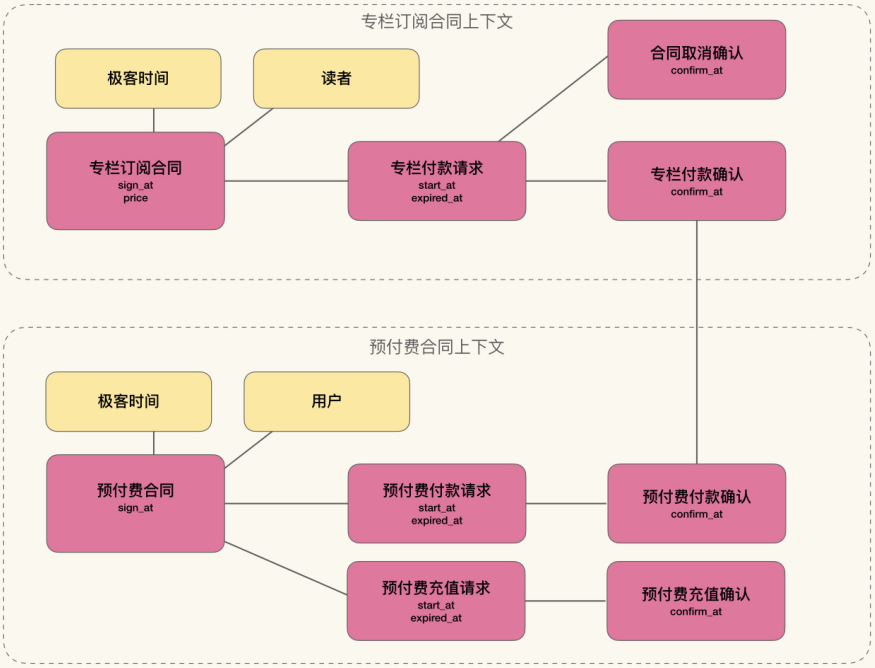
因为在专栏订阅合同上下文的履约确认中，我们关联了另一个合同，也就是移动支付协议上下文中产生的凭证，而这个凭证是由合同上下文中的义务方移动支付供应商提供的。

那么这种跨合同上下文的凭证引用，实际上就表示了不同合同履约项中权利方与义务方间的协作。专栏付款确认引用了移动支付确认作为凭证，也就是说，专栏订阅合同上下文中，专栏付款履约项的权利方极客时间，和移动支付协议上下文中，移动支付履约项的义务方移动支付供应商，共同协作，为用户（移动支付协议合同上下文）完成了支付，以帮助读者（专栏订阅合同上下文）完成了订阅。

说句题外话，你是不是觉得不同的合同上下文和服务边界很像呢？有点儿眼熟对吧？**这其实就是服务边界在业务上的体现。我们费尽心力，希望从聚合和其他技术能力上寻找的服务边界，天然就存在于业务中。而且具有明显的结构特征！只要睁眼看一看，怎么还能找不到服务边界呢！**

好了，言归正传。我们通过跨合同上下文间的凭证引用，建模了不同服务之间的协作。但是**对其他上下文中凭证的引用，实际上表示了不同上下文间的依赖关系。**

也就是说，在上面的例子里，专栏订阅合同是依赖于移动支付协议的。那么如果我们引入另一种支付模式，比如说预充值抵扣，那么我们可能就会引入另外一个合同上下文：预付费合同。而专栏订阅合同也会依赖于它。如下所示：

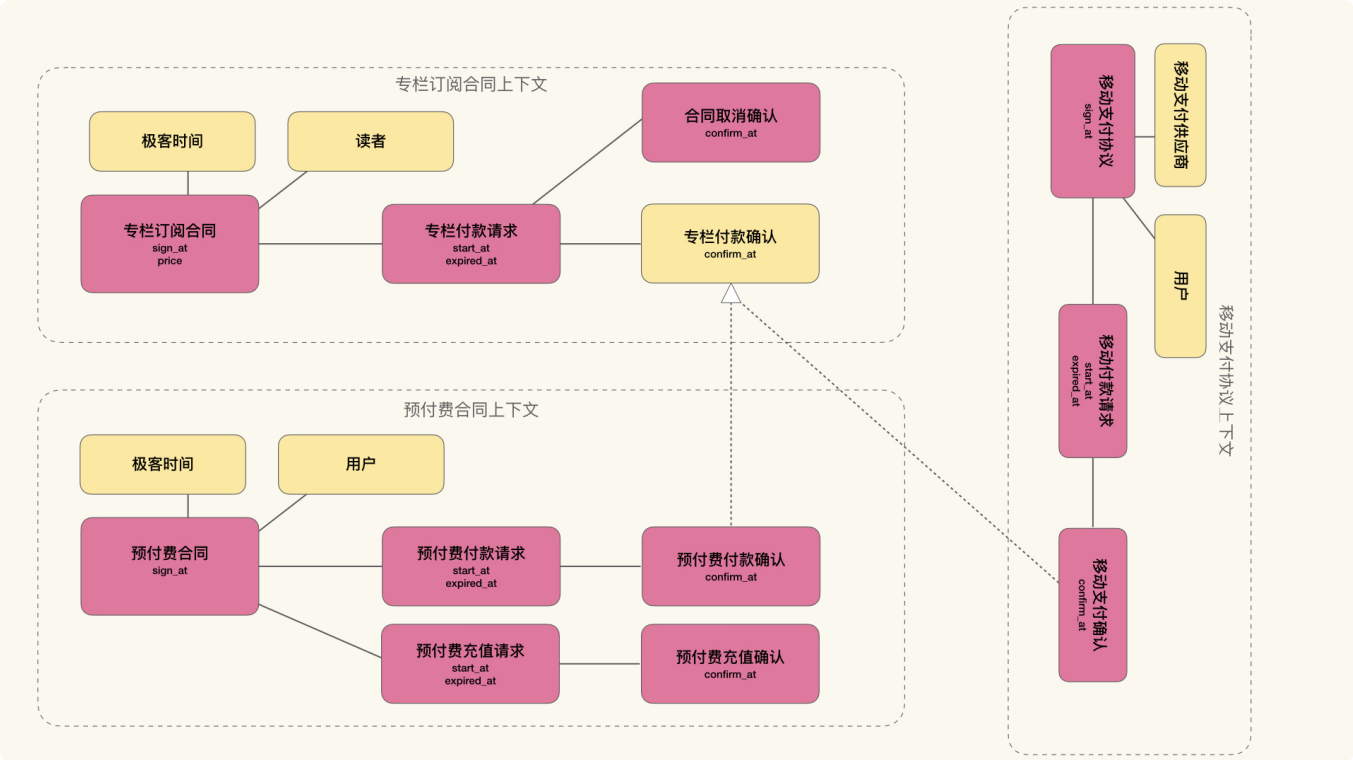


这里明显会出现一个坏味道，随着新支付方式和手段的引入，专栏订阅合同上下文会不断地去依赖新的合同上下文。

且不说由于依赖关系引起的变化传播，单从业务上讲，专栏订阅合同是极客时间的核心业务逻辑，也是核心差异化所在。而不同的支付合同，更像是为了支撑这个核心业务逻辑而存在的。那么对于这种依赖关系，我们就需要小心了，因为让核心逻辑依赖于支撑逻辑，总是一种坏味道。

业务系统中的变化点

不过解决的办法也很简单，那就是**反转依赖**，我们可以让**履约确认角色化，让其他合同中的凭证来扮演这个履约确认。**这其实就是我们在第 9 节讲的能力供应商模式，它是面向对象技术中反转依赖的一个小应用。如下图所示：



通过将专栏付款确认角色化，我们就在专栏订阅合同中引入了一个变化点。就意味着这个履约项，可由多种不同凭证来确定，也就是存在跟其他不同合同上下文交互的可能性。**而这种对业务变化的判断，并不是源自技术方案，而是从业务本身的结构出发，寻找可能存在的变化点。**

于是我又要说句题外话了，**我们不停地通过各种建模手段，希望可以预判业务改变的方向，尽早做出隔离，从而更好地响应业务的改变。同样的，业务逻辑变化的可能性天然也就存在于业务中，而且具有明显的结构特征！只要睁眼看一看，怎么能找不到业务变化点呢！**

小结

你肯定还记得我们在第 13 节讲云计算给建模带来的挑战时，总共讲了四条：

确立一种模型结构以反映弹性边界；

在弹性边界切分业务上下文时，维护业务一致性；

从异步模型的视角，解读业务逻辑；

在异步调用产生中间态异常时，维护业务一致性。

我们来看看怎么应对这四个挑战。**前两条与弹性边界有关**。我们知道合同上下文表示了服务边界，其中合同的乙方就是服务提供者，甲方就是消费者。那么是不是合同上下文就是弹性边界呢？

显然不是！因为在合同中，不同的履约项明显具有不同的弹性诉求。就以专栏订阅合同为例，付费内容访问明显比专栏支付与断更补偿具有更多的容量诉求。

所以**合同上下文并不是弹性边界，履约上下文才是弹性边界**。此外，由于我们分离了领域逻辑和业务逻辑，那么**领域逻辑也是弹性边界**。

在领域逻辑中，数据一致性为主导；而在业务逻辑中，是合同上下文中对业务逻辑的聚合为主导。合同上下文中的业务一致性，也就是履约与违约构成的一致性，因而业务逻辑的弹性改变并不会影响领域逻辑。

所以我们说，**通过切分领域与业务边界，分别以领域逻辑边界和合同上下文中的履约上下文作为弹性边界，就能解决在弹性边界切分业务上下文时的一致性维护问题**。

后两条与异步模式有关。这里有意思的是，如果我们完全从业务角度出发，遵循主张 - 履约的方式建模（比如用 8X Flow 建模），这根本不是问题：因为业务本身就应该通过异步解读；异常状态作为违约，触发其他履约项或诉诸法律。

也就是说，我们之所以会认为后两条是问题，主要是因为我们采用了技术的出发点，从解决方案去理解问题，才有了如此结果，这是我们应该反思的。

至此，我们不仅通过 8X Flow 获得了一种纯粹的从业务角度出发的建模方法，而且，通过 8X Flow 得到的模型，可以帮助我们解决云时代给业务建模带来的四个挑战。

最后讲一个小题外话。**表达想法、提出问题，问题就等于解决了一半。非常期待你能把自己的想法或问题、总结或笔记分享出来，我在留言区等着你**。

思考题

对于其他合同上下文的引用，能不能将其转化为领域系统（比如支付子系统），并与业务部分整合？我们现在的建模是从合同确立开始的，那么合同确立之前要怎么建模？

分享给需要的人，Ta订阅后你可得 20 元现金奖励

赞 4 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 14 | 8X Flow（上）：何为业务？何为领域？

下一篇 16 | 8X Flow（下）：多于一个例子

精选留言 (10)

写留言

菜菜
2021-08-06

难道只有我一个人现在似懂非懂吗？

展开

2

bird
2021-08-05

听过一门财务课说财务是所有业务的共同语言，任何一家公司的核心业务都能通过三大报

表呈现出来。这里说的“合同履约”或者财务上的很多概念思想是可以作为业务建模的“统一语言”的吧

作者回复: 对

2

jack
2021-08-05

学习了，如果分析的只是，企业内部业务活动，比如专栏制作流程，是不是就不适用了，或者这本身就不是业务建模的一部分？

展开

作者回复: 下一课

1

顿晓
2021-08-11

对于其他合同上下文的引用，能不能将其转化为领域系统（比如支付子系统），并与业务部分整合？
转换为领域系统，当做普通凭证被合同上下文引用，是不是进一步简化了业务建模？少了一个支付上下文。

shen
2021-08-10

要多看几遍，抽象有点高，可能我的功底不够

展开

Janson
2021-08-09

感悟："合同"作为一种统一语言，用于描述权责方基于这个凭证的协作，所以可以看作业务上最小上下文，例如订单（买家、卖家）、例如快递单（卖家、快递公司）等。

思考题：1. 对于支付系统，要从公司的业务来看，普通的电商公司支付就是支撑域，像支付宝或者微信对外提供服务能力（包含运营能力）的就可以做成核心域。2. 在合同确立...

展开

Oops!
2021-08-07

是否可以这么认为，本文认为，涉及合同/履约的（和财务相关的）才算业务，其他都算领域？有个困惑，大部分业务逻辑中都不会非常明显的和合同直接相关，很多逻辑虽然涉及权责，但也没有明显的合同概念，这些在业务建模的时候如何处理呢？比如，专栏转让这个流程，实际操作中并没有显式的合同出现，这时候加一个“转让合同”这样的三方合同会不会有点硬拗的感觉？另外，很多内部流程并不直接涉及合同和履约，比如仓库内的...

展开

作者回复: 请回到现实中看。在想软件是怎么会之前，看看现实是怎么回事

姚磊
2021-08-06

很多内部系统都看不到合同，逻辑也非常复杂，互相纠缠，该怎么建模？
另外，在例子中，怎么发现采购合同的，用户和商家也没有签订类似的合同吧

作者回复: 订单就是采购合同

狩月
2021-08-06

"通过将专栏付款确认角色化，我们就在专栏订阅合同中引入了一个变化点。"

从建模的角度反推，是否变化点通常都可以由时标对象的角色化来得到？如果做不到这个过程， 意味着之前的建模可能有问题？

展开

作者回复: 并不是时标 而是履约确认

梦倚栏杆
2021-08-06

6666，大开眼界，从没想过还可以这样操作

展开